D

⑲日本国特許庁（ＪＰ）　　⑪特許出願公開

⑫公開特許公報（Ａ）　　昭63－64178

⑤Int.Cl.⁴	識別記号	庁内整理番号	
G 06 F 15/66		8419－5B	㊸公開　昭和63年(1988) 3月22日
15/16	370	T－2116－5B	

審査請求　有　　発明の数　1　（全26頁）

㊿発明の名称　画像処理システム

㉑特　願　昭62－132016

㉒出　願　昭62(1987) 5月29日

優先権主張　㉜1986年8月29日㉝米国(ＵＳ)㉛902343

⑫発明者　ハングウエン・リ　アメリカ合衆国ニューヨーク州プレザントビル、デイアフイールド・レーン・サウス60番地

㉑出願人　インターナショナル・ビジネス・マシーンズ・コーポレーション　アメリカ合衆国10504、ニューヨーク州　アーモンク（番地なし）

㉔代理人　弁理士　岡田　次生　外1名

明　細　書

1．発明の名称　画像処理システム

2．特許請求の範囲

(1) システム全体を制御するホストコンピュータと複数の基本演算器のアレイとを備え、前記基本演算器は、

(a) メモリと、

(b) 前記メモリに接続されたＡＬＵと、

(c) 内部短絡経路及び他の基本演算器との間の外部連結の形成を前記ホストコンピュータから与えられる連結制御パターンに従って制御する連結制御機構と、

を有することを特徴とする画像処理システム。

(2) 特許請求の範囲第(1)項記載の画像処理システムにおいて、前記ホストコンピュータから与えられる連結制御パターンは前記連結制御機構に備えられた複数のパターンレジスタ内に順次保持されるとともに、連結制御機構に備えられたパターン選択レジスタにより連結制御パターンの切換えが直ちに行なわれるよう構成されていることを特徴とする画像処理システム。

3．発明の詳細な説明

以下の順序で本発明を説明する。

Ａ．産業上の利用分野

Ｂ．従来技術

Ｃ．解決しようとする問題点

Ｄ．問題点を解決するための手段

Ｅ．実施例

Ｅ１．実施例のハードウェア構成

Ｅ２．多形網構造

Ｅ３．制御アルゴリズム

Ｅ４．アプリケーション

Ｅ５．結論

Ａ．産業上の利用分野

本発明は、基本演算器（プロセッシング・エレメント：ＰＥ）のネットワーク（アレイ）から成るアレイプロセッサを備えた画像処理装置（並列計算機でもある。）に係り、更に詳しくは、各基本演算器が他の基本演算器とどのように連結する

かを制御する連結制御機構を備えることによりネットワークの構造がプログラムに従って選択できるような画像処理装置に関する。

B．従来技術

以下に示す刊行文書及び米国特許が技術技術の代表的なものである。

(1) Sternbergの"生物医学的画像処理(Biomedical image processing)" Computer、Jan. 1983、には、近隣のディジタル画像データをくり返し処理していくための、各セルが近接のセルに連結されて成るセルアレイが示されている。曲がりくねったシフトレジスタが3×3の近傍セルへの入力を可能にしている。

(2) Turney等の"部分的に重った個所の認識(Recogmjing Partially Occluded Parts" IEEE Transactions on Pattern Analysis and Machine Intelligence、July、1985、pp. 410-421には、テンプレートの一致に重点を置きながら、ホフ変換を行う種々の技法が示されている。

(3) Mudge等の"画像データの並列処理における特徴依存アルゴリズムの効率(Efficiency of Feature Dependent Algorithms for the Parallel Processing of Images)" IEEE 0190-3918/83/0000/0369 1983、369-373には、基本演算器が連絡ネットワークを通じて連絡するような重多プロセッサの利点について示されている。著作は特徴依存アルゴリズムと特徴非依存アルゴリズムとの相違について研究している。

(4) Sternberg等の"工業的多形性(Industrial Morphology)"には、単一システム内における画像処理とパターン認識の結合を示している。

(5) D. E. Shawの"NO-VONスーパーコンピュータ(The NON-VON Supercomputer)"内部レポート、Columbia University、Aug. 1982には各基本演算器がI/Oスイッチを備え、個々の基本演算器を活動化、非活動化させるフラグレジスタを有するような大型並列処理システムが示されている。

(6) M. J. Kimmel、R. S. Jaffe、J. R. Mancleville、及びM. A. Lavinの"多形画像変換計算機、近傍プロセッサのパイプラインの再構成アーキテクチャ(MITE：Morphic Image Transform Engine、An Architecture for Reconfigurable Pipelines of Neighborhood Processors" IBM RC 11438、Oct. 10、1985には、オペレータ制御下でバス接続を通じてPEとPEとが種々の相互接続を行うようなPEの再構成可能なネットワークが示されている。

(7) A. J. Kessler及びJ. H. Patelの"障害に対して冗長性を有する再構成可能な並列パイプライン(Reconfigurable Parallel Pipelines for Fault To lerance)"、IEEE、CH1813-15/82/0000/0118、1982には再構成可能なパイプライン接続が示されている。

(8) S. R. Sternbergの"画像処理用並列アーキテクチャ(Parallel Architecture for Image Processing)"、IEEE、CH1515-6/79/0000-0712、1979には完全接続されたPEネットワークが示されている。

(9) T. N. Mudge、E. J. Delp、L. J. Siegel、及びH. J. Siegelの"マルチマイクロプロセッサシステムを用いた画像データのコード化(Image Coding Using the Multimicroprocessor System PASM)" IEEE、82CH1761-6/82/0000/0200、1982、には相互接続ネットワークにより相互接続されたPEが示されている。

(10) S. R. Sternbergの"並列画像処理用の言語及びアーキテクチャ(Language and Architecture for Parallel Image Processing)"、<u>Recognition in Practice</u>、North-Holland Publishing Co., 1980には複雑なPEネットワーク及び動作説明が示されている。

(11) 米国特許第4174514号(1979年11月13日)には、2つの隣接する画像スライスの重なり合う領域の画像データへの中間アクセスのための互いに相互接続された隣接するPEより成るアレイプロセッサが示されている。

(12) 米国特許第4215401号(1980年7月29日)「細胞型ディジタルアレイプロセッ

サ」には、直交2軸に沿って周りの4つのPE（細胞）と連結するPEより成るアレイプロセッサが示されている。

(13) 米国特許第4380046号（1983年4月12日）「大型並列処理計算機」には、隣接するPEのPレジスタからPレジスタあるいは他のGレジスタへと縦方向あるいは横方向に沿ってデータビットを空間的にシフト（スライド）させるような画像処理装置が示されている。各PEは算術論理循環ユニット（ALRU）、I／Oユニット、及びローカルメモリユニット（LMU）を有している。ALRUは、バイナリーカウンタ／シフトレジスタサブユニット、ロジックスライダサブユニット（Pレジスタ）、及びマスクサブユニット（Gレジスタ）という3つのサブユニットから構成されている。

(14) 米国特許第4398176号（1983年8月9日）「データ／命令共通バスを有するデータ分析装置」には、バス上の情報がデータとして用いられるべきか命令として用いられるべきかを制御する外部共通制御ラインに各PEが接続されている画像処理アレイが示されている。

(15) 米国特許第4601055号（1986年7月15日）「画像処理装置」には、像の1つ1つに対応した低レベルの画像処理装置をピクセル毎に正逆方向に1つづつ変換する装置が示されている。

従来技術によれば、パイプライン型やその他の型内でのPEの固定的な構成については示されている。従来技術によれば、スイッチングネットワーク及びバスを通してPEのネットワークを再構成することについては示されている。しかしながら、従来技術には、多形（変形）網のような非常に高速にスイッチングを行う技術については何らの示唆もない。多形網はプログラムにより再構成させることができ、紐状、ツリー状、キューブ状、ピラミッド状というように、処理内容に応じて適切な構成とすることができる。

セル構造オートマトンは画像処理、画像認識等に非常に有用であることが認められている。しかしながら、現在のオートマトン用のネットワーク構造は紐状なら紐状のまま、メッシュならメッシュのまま、ツリーならツリーのまま、キューブならキューブのまま、ピラミッドならピラミッドのまま、というように固定された構成である。各構成は特定の計算については適しているが、その構成に適さない計算も存することになる。ネットワーク構造が固定的に作り付けられているので、不適な計算であることが判った場合にも、そのネットワーク構造は変えられず、低効率のまま計算を行なわざるを得ないことになる。

例えば、N×Nの網目構造が画像処理でのローカルオペレーション（特殊な操作）に適していても、グローバルオペレーション（一般的な操作）には不適である。N×Nの網目構造では最小値計算（MINIMUM）にNサイクルを要するが、ツリー構造であればlogNサイクルで済む。だが、ツリー構造の場合には、近傍の基本演算器との接続が不足しているため、画像処理用のローカルオペレーションには不適なのである。

基本演算器の相互接続構造がアルゴリズムに不適なときには計算効率も低いが、その上に、構造が固定的なときには、用るアルゴリズムの設計に制約が生じる。これは、基本演算器の相互接続構造がデータの流れに対する制約を決定するからである。例えば、紐状構造では、データの流れは左から右というように一方向であり、このような構造では、紐状にデータが流れる場合に適したアルゴリズムに制限される。このように、固定したネットワーク構造は、各々が特別の適用性を有してはいるが、適用範囲自体は狭いものである。

固定的な相互接続パターンの欠点は、画像処理及び中間レベルの処理（画像情報を記号情報に変換する処理）を同時に効率良く実行することができないということである。このような欠点は、セルオートマトンであるコンピュータビジョンにとって特に重大であり、そこでは画像処理と中間処理とが不可欠部分である。この欠点は入出力問題にとって深刻である。というのは、画像処理後の画像データを後続の更なる中間処理のために出力

しなければならないからである。

C. 解決しようとする問題点

本発明の目的は、プロセッサアレイ中の基本演算器（プロセッサエレメント：PE）の各々をプログラム制御により高速に連結し、PEを、全方向（多方向）連結の場合に要求されるようなコスト高を招くことなく、効率良く組み合せることである。

他の目的は、PEの再組み合せを、時に応じて、再組み合せの必要性を検知したコンピュータの制御下により、及び、再組み合せの必要性を予見したオペレータの操作により、実行することである。

更に他の目的は、外部メモリデータをPEに接続することにより、必要なハードウェアを軽減するとともに操作の柔軟性を向上させることである。

D. 問題点を解決するための手段

本発明の特徴は多形（変形）網目型の基本演算器（PE）を用いることである。各PEは、メモリを備えたALUにより処理能力を備え、地理的な接続即ち論理的な接続関係をプログラム制御する能力を備えている。

他の特徴はPEがプログラム制御下で短絡能力を有することである。この短絡能力により、送り元のPEと遠く離れた送り先のPEとの間の一連の中間に存在するPEは単に導電線としてのみの役割を果すだけとなり遅延サイクルを生じさせることがない。

他の特徴は多形網目ネットワーク構造を有することであり、このネットワーク構造は各PEの外部の通常の網目と各PEの内部の内部ネットワークとの組み合せネットワークであり、ソフトウェア制御を通じて通常のパターンと他の新しい有用なパターンとが計算処理に適したものとなるように調整されるようになっている。多形性という特徴により、ネットワーク自体が再構成され、柔軟性の有るアルゴリズムの設計と広範なアプリケーションに用いることが可能となる。

画像処理に関連しては、多形網目構造により、中間レベル処理（画像データの記号データへの変換処理）が効率良く実行され、入出力問題に関する一連の問題を解決できる。

他の特徴はPE内にフラグレジスタを有することであり、フラグレジスタは再構成操作のような条件付操作のために用いられる。

他の特徴は複数ビットパターンレジスタを所定数有することであり、所定数のソフトウェアにアクセスすることによりハードウェアパターンを選択でき、各パターンは多形網目により実現され得る。これらのパターンにはバス構造、幾つかのツリー、キューブ、ピラミッドが含まれ、各パターンは関連する計算のタイプに適しており、複数のパターンレジスタのうちの選択された1つのパターンレジスタ内の1種のビットパターンに応答してクロスバースイッチにより選択され得る。

本発明の利点は比較的安価でありながらスループットスピードが高いということであり、これはPE内のプログラム制御される連結能力によるものであり、こうして、物理的にも電気的にもプロセッサアレイの適正化がなされる。

他の利点は中間レベル処理の結果がプログラムと結合して効率良く適正な再構成化に役立つことである。即ち、中間データは適正な再組み合せ機能を呼び出すために用いられる。

他の利点は構造が簡単で小さな面積上に構築でき、非常に多数の相互連結されたPEを少ない数のチップで構成できるという事である。

E. 実施例

E1. 実施例のハードウェア構成

第1図はM×Mのアレイ1を有する画像処理システムを示し、アレイ1を構成する多形網目基本演算器（PE）2はホストコンピュータ3により制御されるようになっている。各PE2は連結子を有し、この実施例では、4つの連結子を有し、各連結子は隣りの連結子と直交している。これら直交する連結子には方向識別子NESW（北東南西）が付されている。これら連結子NESWの役割は隣接するPE（4つ存在する。）に対して出力を直接に与えることである。全般的なプログラミング制御とハウスキーピング制御はホストコンピュータ3により行う。

第１図にはＰＥ２の一層詳しい内部構成も示されている。ＰＥ２は、ＡＬＵ６、メモリ（ＭＥＭ）７、連結子制御機構（ＣＣＭ）８、及び４つの連結子Ｎ、Ｅ、Ｓ、Ｗを有し、これらの全体的なプログラミング制御とハウスキーピング制御はバス９を介してホストコンピュータ３が行うようになっている。

４つの連結子ＮＥＳＷを有するＰＥ２は直交座標上に配されており、この簡単な網目構造は画像処理において有利である。また、この簡単な網目構造はＶＬＳＩ上に形成する上でも有利である。

直交座標上にないＰＥ２同士は直接的には配線で結ばれていない。対角線上のＰＥ２同士の連結や遠く離れたＰＥ２同士の連結は配線上望ましくないものであり、製造上の困難を伴う。多数配線の束は（静電）容量と（信号経路）長さとを有し、そのために固有の信号遅延をもたらす。

一実施例では、直交座標上にない関係のＰＥ２同士は、それらの間に介在する直交座標上のＰＥ２を介して、各々のＣＣＭ８の働きの下で、互いに接続される。ＣＣＭ８が制御する連結子のパターンは、入力側（送り先側）ＰＥと出力側（送り元側）ＰＥとを効率良く短絡させるようになっている。ＰＥの連結経路はチェスの城将（将棋の飛車に相当する。）のように直交軸方向に沿った複数個の長さ或いはチェスのナイトのように直交軸の両方向に沿った複数個の長さに相当するものであり、チェスのビショップ（将棋の角に相当する。）のように対角線方向に延びるものではない。複雑な経路も設定できるし、信号遅延を生ずることなく多数のＰＥ２を経由する非直交座標上の関係のＰＥ間をつなぐ経路が設定され得る。

簡単な直交座標に沿った連結状態に代えて、対角線上のＰＥ２相互を直接的に連結してもよいが、極めて遠く離れたＰＥ２相互を直接的に連結することは、そのような連結状態は簡単な直交座標上の或いは対角線上のＰＥ２の組み合せをプログラム制御しても達成されることであり、好ましいものではない。

第２図は第１図中のＣＣＭ８の動作を説明するためのものであり、実際の内部ハードウェアを示そうとするものではない。ＣＣＭ８の重要な機能は、パターンレジスタ内のビット値の制御下でＸ方向の横棒の連結子ＮＥＳＷのどれかをＹ方向の横棒の連結子ＮＥＳＷのどれかに連結させるというスイッチングネットワークとしての機能である。図示の例では、２つのパターンレジスタ２１、２２が選択的に用いられ、その選択はパターン選択レジスタ２３の制御下で行なわれている。この例では、たとえ全てを連結させることが実際には可能であっても、４×４のマトリックス内の全ての交点を連結させることが必要なわけではない。マトリックス１０はＹ軸方向に沿った複数の入力線１１を有し、これらはＸ軸方向の導線Ｓ１２に接続している。図示のように、Ｎ連結子１３は、連結用電子部品１４、１５及び交線１６を介して、Ｅ連結子１７に連結されるよう設定されている。他のＳ連結子２７、Ｗ連結子２８、Ｎ連結子２９への連結が交代的に或いは同時的に行なわれてもよい。制御は１６ビットのパターンレジスタ２１、２２により行なわれる。連結線２４、２５、及び２６への連結も可能である。交線スイッチ１０はパターンレジスタ２１および２２のうちの選択された方のレジスタ内のビット値により制御されて１６個の交点の１つ又は複数を連結させる。

各ＰＥ２はパターンレジスタ２１、２２の一方のビットパターンに指示されて隣接する１つ又は複数のＰＥ２と連結する。パターンレジスタ２１、２２を切換えることにより連結関係の変更が直ちに行なわれる。説明上、パターンレジスタ２１、２２は標準的若しくは変形的なパターンレジスタであり、パターン選択レジスタ２３は０か１かの２値レジスタであるとする。

第３図は第１図のＣＣＭ８の内部の機能ブロックを示し、これらの機構ブロックが第２図に関連して説明したところの動作を実行する。簡単な論理連結機構３０はＡＮＤ、ＯＲ、ＸＯＲ（排他的ＯＲ）、及びＡＮＤＡＬＬＢＩＴ機能を有する。他の機構でもよいが、複雑なものであればコスト高を招く。論理連結機構３０は、第１フラグ（Ｆ

1）３１及び第２フラグ（Ｆ２）３２内のフラグ入力を受信する。これらフラグ入力は機構３０による制御を変更させる。シフトレジスタマスク（ＳＲＭ）３３は真出力（ＳＲＭ）３４及び相補出力（ＳＲＭ）３５を出力し、これらの出力信号は、機構３０を制御するＸレジスタ用シフトレジスタ（ＳＲＸ）３７及びＹレジスタ用シフトレジスタ（ＳＲＹ）３９のビット値を選択するために用いられる。

Ｘレジスタ３６及びシフトレジスタＸ（ＳＲＸ）３７はＸ方向の隣接するＰＥ２への論理的、従って幾何的な連結状態を制御する。Ｙレジスタ３８及びシフトレジスタＹ（ＳＲＹ）３９はＹ方向の論理的即ち幾何的な連結状態を制御する。通常、Ｘレジスタ３６及びＹレジスタ３８には、画像処理システム内でのＰＥ２の直交座標の値が入っている。シフトレジスタＸ（ＳＲＸ）３７及びシフトレジスタＹ（ＳＲＹ）３９は、シフトレジスタマスク（ＳＲＭ）３３と共働して、Ｘ及びＹ方向の夫々の一連のビットグループを誘導するために用いられる。第３図のＣＣＭ８の詳細な機能は多形網目により１２の異なるパターンを形成する１２の例により説明される。

各ＰＥ２は、そのＰＥ２に対して示されたビット値に基づいて算術論理演算操作あるいは論理変換操作を実行し、あるいは、何らの操作も実行しない（ノー・オペレーション）。このような操作の実行あるいは不実行に加えて、ＰＥ２は周りのＰＥ２のうちの選択されたものと連結される。ここで言うところの隣接するＰＥへの連結は短絡と呼べるようなものである。短絡は瞬時に起こる。この短絡による連結の速度は、通常の操作（ノー・オペレーションを含む。）のように１サイクルの遅延をもたらすものではなく、電気的速さ、即ち、光の速さである。したがって、遅延サイクルを生じさせることなく複数のＰＥを経由して遠隔のＰＥと連結することが可能となる。また、ジグザグ経路により同一の列にも同一の行にもない離れたＰＥ２と連結できる。こうして、対角線方向の連結子がなくとも、対角線方向に隣接するＰＥ２への、あるいは対角線方向に遠く離れたＰＥ２への更には、対角線上から外れた遠いＰＥ２への連結が可能である。

本発明による多形網目画像処理システムは、適正にプログラム制御されることにより、ＰＥ間の複雑な相互連結を可能にする。各ＰＥは自分に与えられた情報に準拠して適正な算術論理操作、変換操作、あるいはノー・オペレーションを実行するが、そうでないときには信号遅延のない短絡的連結を行うことができる。

どれ程の数の複雑な連結ができるかは、パターンレジスタ内のビット値の数に依り、パターンレジスタの数やＣＣＭの構成にも依る。しかし、多数のＰＥを必要とするような複雑な連結はコスト高となる。そこで、好ましい実施例では、適切な連結状態の範囲を限定した。これらについては、第４図ないし第２０図に示されている。

第４図は線形な配列の連結状態を示している。西（Ｗ）から東（Ｅ）への線形的配列４１と北（Ｎ）から南（Ｓ）への線形的配列４２が存在している。

第５図及び第６図は行方向のツリー配列についての２つの連結状態を示している。そのうち、第５図にはストリング（紐）４５～５１が示され、これらＸ方向に沿ったストリング４５～５１は種々の長さを有している。第６図にはもっとツリー構造らし連結状態が示され、ここでは、７つの入力が４つのステップを経て１つのＰＥ５２へと導かれている。

第７図ないし第１０図（第７図が本発明を示し、第８～１０図は従来の例を示している。）は本発明の多形網目型ＰＥを用いた結果、チップ上の面積及びＰＥ間の接続に要する配線長さがいかに改良されたかを示している。７図の多形網目６１のチップ面積は、第９図の直角配線型ツリー構造のパターン６２と比較して極めて小さい。第８図のパターン６３の３×３のウィンドの平均的接続（配線）長さは第７図の場合の１．５倍であり、第１０図のパターン６４の３×３のウィンドでは２．１２５倍である。

第１１図は逆ツリー構造を示している。第１１図は第５図と非常に似ているが、第５図のツリー構造ではＰＥから情報を（１ヶ所に）集めているのに対し、第１１図の逆ツリー構造では情報を広めている。

第１２図ないし第２０図には各々、多形網目ＰＥ間のプログラム可能な相互連結状態のいくつかの例が示されている。これらの図において、四角で囲まれた丸印は何らかの中間処理を行うＰＥを示し、単なる丸印は単なる通過用（短絡用）に用いられるＰＥを示している。ただし、どちらの場合のＰＥもプログラムにより制御されている。各パターン（ＰＥの相互連結状態）は１６ビット値で表わされ適当なパターンレジスタ内に収容される。

第２１図は本発明による多形網目ＰＥの好ましい一実施例の内部構成を示している。図中、破線９３で囲まれた部分は、種々の実施例を通じて比較的共通的な構造である。ＡＬＵ９６は出力１及び出力２を有するとともに、入力側にはマルチプレクサを備えている。入力は入力端子９４及びレジスタＮＥＳＷ９５から与えられ、このレジスタＮＥＳＷ９５には第１メモリ（Ｍ１）及び第２メモリ（Ｍ２）、あるいはＡＬＵ９６の出力１及び出力２からデータが与えられる。ローカルメモリ１０１はＡＬＵに適切な計算及びハウスキーピングを行なわせるために用いられる。外部メモリデータ線（ＥＭＤ）１０２からは外部メモリ（図示せず）からデータが供給され、マルチプレクサ１０３を経て、ローカルメモリ１０１とともに、第１メモリ（Ｍ１）及び第２メモリ（Ｍ２）に接続されている。ＡＬＵ９６の出力１及び出力２は同じＰＥ９３に供給されるともに、ＣＣＭで選択された他のＰＥ９３にも供給され得る。

外部メモリデータ線（ＥＭＤ）及びＭ１、Ｍ２につながる複数のバスにより、外部メモリデータと内部メモリ（ローカルメモリ１０１）が接続され、ＰＥ９３はローカルメモリ１０１及び外部メモリと協働することになる。

外部メモリデータ線（ＥＭＤ）１０２により個々のＰＥはホストコンピュータ（Ｈ）３内のメモリ或いは独立型のメモリに直接的に接続される。このような接続は従来のアレイプロセッサには見られないものであり、このような接続により第３図に示した機能が実現される。

第３図に示したような方向制御及び論理機能制御下におけるスイッチング機能は外部メモリ及びＥＭＤ１０２に関連して実行される。第３図に示したスイッチング機能を実行するための手段については、その全ての部分を各ＰＥ内に持たせてもよいし、あるいは、その全ての部分をＰＥ外のものとしてもよい。ただし、後者の場合には、得られた条件についてはＥＭＤ１０２を通じてパターンレジスタ選択用レジスタＲｐ９９に送ることとする。

２つのパターンレジスタＰＲＯ９７及びＲＲＩ９８により命令サイクルの遅延なしで１つの連結パターンから他の連結パターンへと瞬時にスイッチングすることができる。この瞬時のスイッチングは１ビットレジスタのパターンレジスタ選択用レジスタ（Ｒｐ９９）により制御される。２つのレジスタ９７及び９８のうちの一方のレジスタ内の値がもはや使用する必要のないものとなったときには、そのパターンレジスタには新しいパターンがロード（転送）され得るようになっている。このようなロードは自由に行なわれる。即ち、ＡＬＵ操作が行なわれている時に同時に行なわれ得る。なお、画像処理システムでは通常、ＰＥは１ビット処理を行い、多くの場合、比較的簡単な構造である。１６ビットパターンをパターンレジスタ９７及び９８にロードすることは比較して述べればやや面倒な作業である。パターン選択用レジスタ９９にもある値をロードしなければならないが、この場合は１ビットであるから簡単である。

パターンレジスタ９７及び９８を書換えるために処理を中止しなければならないような場合がある。この場合、２つのパターンレジスタ９７及び９８の書換えに３２サイクルを要し、選択用レジスタ９９へのロードに３３番目のサイクルを要する。しかしながら、多くの場合、パターンレジス

タへのロードのために特別の時間が必要になることはない。適当な命令中に、オペレータがレジスタ９７あるいは９８、あるいはレジスタ９９内に１ビットだけロードすることができるからである。ＡＬＵ９６がフル稼動しているときには、ＡＬＵ９６が算術論理演算を行っている最中でも、ＣＣＭ内の１つあるいは全てのレジスタの書換えを行うことができる。このような融通性のあるロード動作は、インストラクションゲート１００がロード機能を実行するように準備されたときに、入力端子９４からＡＬＵ９６のマルチプレクサを通って出力１及び出力２により行なわれる。

パターンレジスタ９７又は９８から選ばれた１６ビット値は４×４クロスバースイッチ１０４（詳細は第２図に示してある。）を細かく制御するためのものである。

ＰＥの典型的操作は短絡操作と処理操作（活動操作）である。短絡操作の場合には、例えばパターンレジスタ９７が短絡操作用に設定されて短絡操作を実行させ、ＰＥを１つ又は複数の他のＰＥに短絡接続させる。ＰＥを処理操作にするときには、パターンレジスタ選択用レジスタ９９を処理操作用に設定して、短絡バイパス用に設定されているパターンレジスタ９７から処理操作用に設定されているパターンレジスタ９８へとスイッチ操作する。

ＰＥの処理操作につき更に説明する。

第１図ないし第３図に示された多形網目はＣＣＭに制御されて複数のパターンに変形することができる。これらのパターンのレパートリーは計算のレパートリーに対応する。

各パターンに対応する制御アルゴリズムについて述べる。全ての制御アルゴリズムは簡単であり、制御アルゴリズムを実行し易いような所望のパターンの配線状態となっているハードウェアを用いていることができる。

ハードウェア機構は全てのパターンの形成を実行し、ＶＬＳＩ上への実装に適している。

多形網目ハードウェアの大きな利点は、多くのアルゴリズムの線形的な計算量〔Ｏ（Ｎ）〕が対数的な計算量〔Ｏ（logＮ）〕に減じることであり、好ましい。したがって、本発明による新規なアーキテクチャによりlogＮ／Ｎのスピードアップが図られ、例えば、１０２４×１０２４のネットワークにおいてはスピードアップは１００である。

特に画像処理分野では、１つのパターンによる画像処理の後に、得られた画像データが直ちにネットワーク外に送出されるということはない。画像情報は更に別のパターン（例えばツリー構造）によって記号（符号）情報に変換されなければならないし例えば１３３より大きな値のピクセルが多数存在する）。多形網目の機能により、データは出力されなくともよく、その結果、Ｉ／Ｏ操作は十分に減小される。（例えば１０２４×１０２４の画像の例では５桁の減小がある。）Ｉ／Ｏ操作の減小による処理速度の向上の効果は種々のスピードアップ効果の中でも最も影響の大きいものである。

もう１つのパターンは対角線方向に拡がるツリーであり、このパターンが多形網目で形成されることにより、Ａｘ＋Ｂｙ＋Ｃの計算を前述の対数的時間内で実行することが容易となる。ただし、Ａ、Ｂ、Ｃは定数で、（ｘ、ｙ）はピクセルの座標であるとする。コンピュータグラフィクス（計算機図形処理）では、凸状多角形を表示させることとは、影をつくり、切り抜き（クリッピング）を行い、球を描き、ヒストグラム平坦化計算を行い、組織マッピング等の処理を行なう上で有用である。また、ラインマスク、バンドマスク、多角形マスクをつくる上で有用である。更に、第１ホッフ（Hough）変換を計算したり、その逆にノイズ画像からラインを検出したり、あるいはその他の応用において有用である。

有用な１２のセル構造オートマトンと１２のアルゴリズムをつくり出すハードウェア機構を有している好ましい実施例では、ソフトウェア制御下で多形網目を再構成する。以下では、多形ハードウェア及び制御アルゴリズムについて述べる。

多形網目の注目すべき特徴は処理条件により自

らを再構成する能力を有することである。既述のように、ALUの操作によりパターンレジスタ97及び98にはあるビットパターンがロードされ、その結果、各PEはPパターン（P1、…Pp）を有する。

PEの条件Cに適合する再構成により各PEはパターンPiを機能Cとして扱う。まず、全てのPEはPiというパターンから始まり、処理は最初のパターンにもとづいて始まる。処理が進むにつれ、各PEはそのローカル条件を検知し（例えば、これは値が寄り集まることのテストであり得る。）そのパターンを継続するか新しいパターンに置換するかを決める。

条件はグローバル（全体に影響するもの）であってもよく、これはホストコンピュータが各PE条件を一括的に検知し、パターン選択を行い、全てのPEにそれをフィードバックするものでもよい。条件はローカル（一部にのみ影響するもの）であってもよい。新しいパターンの個々のPEに与えてもよいし、適当なPEのグループに与えてもよい。

新しいパターンの選択は幾つかの方法で行なわれる。

(1) 1つには、予め計画されている場合である。一連のパターン（例えば、P1、P2、…Pp）が計画され、この優先順序で利用される。新しいパターンの必要性が検出されると、（P1を用いているとして）次のパターンPi+1が用いられる。そしてPi+2が使用していないレジスタに同時的にロードされる。(2) 他には、機能Cに基づく場合である。

再構成の例には次のようなものがある。

3×3ウィンドがPiにより形成されてフィルタ操作が行なわれ、指標がフィルタ操作の効率を測る条件Cを示している。ここでの目的はCが満足されないときウィンドサイズをを大きくすることである。これに関して、P2を5×5ウィンド、P3を7×7ウィンド、P5を9×9ウィンド等とすることができる。

再構成の別の例はソフト誤りアプリケーションである。各PE内に誤動作に関する条件Cを設計することは共通的なことである。そのような条件は新しい適切な連結パターンPiを決定するために利用される。

E2. 多形網目構造

第1図に示されるように、多形網目はM×MのPE2のアレイ1から成り、各PE2は4つの連結線（N、E、S、W）を有し、各連結線は隣のPE2に連結している。また、各PE2は、ALU6、MEM7、CCM8を備え、ALU6及びMEM8については画像処理分野でも通常用いられているが、CCM8は本発明に特有のものである。

連結制御機構（CCM）8は4つの連結線の他にALU6及びメモリ（MEM）7との間の入出力線を備えている。ここで、A及びBをCCM8への入力線及び出力線であるとしたとき、どこかの入力線Aをどこかの出力線Bに短絡させる（"SHORT_CIRCUIT"）ことにより連結線NESW間の経路指定が実施される。このとき、入力線A上の信号と出力線B上の信号とは等しくなり、例えば、"SHORT_WE"という命令は連結線（W）24の信号と連結線（E）26の信号とを等しいものにする。

第3図はCCM8の機能を示している。前出の"SHORT_CIRCUIT"の動作は条件付動作であり、その条件はCCM8でつくられる。各CCM8は条件信号を発生するための2つの1ビットフラッグF1 31及びF2 32を有する。また、各CCM8は1つのシフトレジスタ（SRM）33を有し、SRM33は両方向にシフト可能で、相補出力34及び35を出力する。

各CCMは一対のレジスタX36及びY37を有し、レジスタX36はPEの行の位置（0≦X≦M−1）を保持し、レジスタY37はPEの列の位置（0≦Y≦M−1）を保持する。レジスタX36及びレジスタY37は各々シフトレジスタSRX38及びSRY39を備え、これらSRX38及びSRY39には各々レジスタX36及びレジスタY37からデータが転送されるとともに

SRX38及びSRY39は両方向にシスト操作できる。SRX38及びSRY39でシフトされた出力はBSRX及びBSRYである。

“SHORT_CIRCUIT”動作についての条件を発生するために幾つかの機能が設けられている。

LOAD reg value：この機能は命令あるいは記憶内容を通じて“ある値”SRM、F1、あるいはF2に転送することである。

COPYSR reg：この機能はレジスタX、Yの内容をシストレジスタSRX、SRYに複写することである。

AND／OR／XOR reg：この機能は次のどれかを行うことである。

1．XとSRX及びXとSRM（あるいはSRM）についてAND／OR／XORを行う。

2．YとSRY及びYとSRM（あるいはSRM）についてAND／OR／XORを行う。

3．上記1及び2の両方を行う。

ANDALLBIT reg：この機能はレジスタ内の全てのビットを“AND”することである。これにより、レジスタがXであれば条件ビットXANDALLを、レジスタがYであれば条件ビットYANALLを、あるいはこれらの両方の条件ビットをつくり出す。

以上から“SHORT_CIRCUIT”動作はBSRX、BSRY、XANDALL、YANDALL、F1、及びF2の組み合せに基礎を置いていることがわかる。

多形網目の残りの2つのブロック（ALU6及びMEM7）は通常の構成と同様である。メモリ7は1マシンサイクル毎に1ビットをCCMに送ったりCCMから受け取ったりする。ALU6も通常の構成と似てはいるが、BSRX、BSRY、XANDALL及びYANDALLの組み合に応答して“SEND”あるいは“RECEIVE”を選択する点に特徴を有する。

多形網目の12種のパターンをその制御アルゴリズムとともに以下に述べる。

E3．制御アルゴリズム

(a) パターン1（線形配列）

第4図に示されるように、PE（M−1、i）の連結子SをPE（0、i＋1）の連結子Nに連結し、PE（i、M−1）の連結子EをPE（i＋1、0）の連結子Wに連結することによって、M×Mの長さの行（図中横方向）の線形配列及びM×Mの長さの列（図中縦方向）の線形配列が形成される。PE（0、0）の連結子E及びPE（M−1、M−1）の連結子Sは各々、列方向の線形配列の開始端及び終末端であり、PE（0、0）の連結子W及びPE（M−1、M−1）の連結子Eは各々、行方向の線形配列の開始端及び終末端である。

制御アルゴリズムは次のとおりである。

```
LINEAR0
{
    MEM＝W; /*action 1 */
    E＝MEM; /*action 2 */
    MEM＝N; /*action 3 */
    S＝MEM; /*action 4 */
}
```

action1（動作1）は連結子W上のデータをサイクルの終りにメモリに置き、action2（動作2）はメモリのデータをデータサイクルの初めに連結子Eに置く。PE（0、0）の連結子Wに注入されたデータは東（図中左）に向って行進し、Mサイクル後には第1行目の全てのPEがデータで満される。次のMサイクルでは、第2行目の全てのPEが前記データで満されるとともに、第1行目の全てのPEは新たなデータで満たされる。action3（動作3）及びaction4（動作4）は同様の動作を列について行なわせる。

このような線形配列の形成には条件は付かない。

全ＰＥが同じ動作を行い、ＣＣＭは利用されない。

(b) パターン２（行方行ツリー配列）

この場合の制御アルゴリズムは次のようである。

```
ROW－TREE0
{
int t; /*t is time step*/
int pattern identifier; /*column position of
a PE, Process ID0*/
int M, logM; /*M is the side size of the
mesh and logM＝logM*/
int treemask＝1; /*a flag to construct the
tree*/

  for(t＝0; t<logM; t++){
    if(¬treemask)
      {SHORT_WE; DISABLE; }
    if(treemask && pattern identifier<t>)
       E＝MEM;
    if(treemask && pid0<t>)
       MEM＝W;
    treemask＝treemask & pid0<t>;
  }
}
```

第５図には１行が８個のＰＥであるときの例が示されている。ｔ＝０で、全ＰＥの“ツリーマスク”は１であり、全ＰＥは使用可能となり、偶数番目のＰＥは奇数番目のＰＥにデータを送る。この様子が“偶／奇”ＰＥのペア間の矢印により示されている。これがツリーの底部を形成する。

ｔ＝１では、ツリーマスクを調べることにより、pid０<０>＝１（pid０の最下位ビット）のＰＥだけが使用可能になる。なお、pid０はＰＥの列位置を表わすための識別子である。使用禁止（ディスエイブル）のＰＥは通常の丸印で示されており、これらはＰＥ１とＰＥ３とを、及び、ＰＥ５とＰＥ７とを連結する。これは“ＳＨＯＲＴ_ＷＥ”という動作（短絡操作）による。こうして第２レベルのツリーが形成される。

ツリーの最高位レベルはｔ＝２のときの制御により形成される。この段階ではＰＥ３とＰＥ７のみが使用可能であり、残りのＰＥは“ＳＨＯＲＴ”という動作により連結を確立してＷ－Ｅ経路をつくるが何の操作もしない。別言すれば、メモリ６内の値を変更させない。

この制御アルゴリズムを別の観点から示したのが第６図である。第６図ではツリーの各レベルの節にはＰＥの識別用番号が付されている。

ツリーパターンは画面を分割したり、全く別の画面にするような例では極めて便利である。このような例でのアルゴリズムの計算量（ステップ数）は、Ｎを入力データの大きさとしたときに、通常Ｏ＝logＮである。このアルゴリズムは通常、Ｏ（＝logＮ）実行タイムを要し、Ｎ＝１０２４であれば、１０２４：１０で表わされるようなスピードアップがある。この種の重要なアルゴリズムはＭＡＸ（最大値を求める）、ＭＩＮ（最小値を求める）、Ｋ番目の大きさの検出、中間値の検出、等にも用いられる。

ＣＯＭを使うと、制御アルゴリズムはレジスタＸ３６をpid０のために、また、レジスタＳＲＸ３７及びフラグレジスタ（Ｆ１）３１をツリーマスクのために用いる。レジスタＸ３６の内容はＳＲＸ３７に複写され、pid<ｔ>ビットはｔ段階でＢＳＲＸに出力され、ツリーマスクとＡＮＤ操作され、最終条件を誘導する。

(c) パターン３（列ツリー）

行ツリーの場合と似ており、Ｍ列ツリーは、ＰＥの行位置である。pid１を制御子として有し、また、Ｎ－Ｓ経路を連結子として有している多形網目により形成される。制御アルゴリズムは次のようである。

```
COLUMN－TREE0
{
int t; /*t is time step*/
int pid1; /*row position of a PE*/
int M, logM; /M is the side size of the
mesh and logM＝logM*/
```

```
int treemask＝1; /*a flag to construct the
tree*/

  for(t＝0; t<logM; t++){
     if(¬treemask)
        {SHORT_NS; DISABLE; }
     if(treemask && ¬pid1<t>)
         S＝MEM;
     if(treemask && pid0<t>)
         MEM＝N;
     treemask＝treemask & pid0<t>;
  }
}
```

列ツリーはデータを列方向に分配させるのに有用であり、その際には、行ツリーの説明のところで述べたように、従来の線形的な計算量［O(N)］のアルゴリズムから対数的な計算量［O(logN)］のアルゴリズムへと変換される。

ROW_TREEと同様に、COLUMN_TREE制御アルゴリズムではレジスタYをpid1に、SRYをcopyYに、F2をツリーマスクの保持のために用いている。SHORT_NSの条件はtime step1でpid<1>を生じさせるF2及びBSRYのAND処理結果に準拠している。

(d) パターン4（直角線ツリー）

直角線ツリーは分類処理、マトリックス操作、最小幅ツリー操作、FFT、及びその他の図形処理アルゴリズムにとって有用なネットワーク構造である。以下に示すORTH_TREEアルゴリズムにより行ツリーと列ツリーとを結合することによってそのような多形メッシュが形成され得る。

```
ORTH_TREE0
{
int t; /*t is time step*/
int pid0; /*column position of a PE,
Process ID0*/
int pid1; /*row position of a PE*/
int M, logM; /*M is the side size of the
mesh and logM＝logM*/
int hmask＝1, vmask＝1; /*flags to construct
the tree*/

  for(t＝0; t<logM; t++){
  /*cycle 1 */
     if(¬hmask)
        {SHORT_WE; DISABLE; }
     if(hmask && ¬pid0<t>)
         E＝MEM;
     if(hmask && pid0<t>)
         MEM＝W;
     hmask＝hmask & pid0<t>;

  /*cycle 2 */
     if(¬vmask)
        {SHORT_NS; DISABLE; }
     if(vmask && ¬pid1<t>)
         S＝MEM;
     if(vmask && pid1<t>)
         MEM＝N;
     vmask＝vmask & pid1<t>;
  }
}
```

多形網目から直角線ツリーを形成することの大きな利点は、(1)チップ面積の減小化と(2)隣接操作の効率化と2点である。

(1) チップ面積の減少化：網目構造及び直角線ツリー構造をレイアウトするために必要なチップ面積は夫々、O［N^2］及びO［$(N^2)\times(\log N)^2$］である。ここで、Nは網目構造の横幅であり、ツリーのリーフの数である。これは、$(\log N)^2$の分だけ節約がなされていることを示し、N＝1024であるとき、多形網目を用いると、そのチップ面は直角線ツリー（第9図参照）の場合の1／100になる。

(2) 隣接操作の効率化：直角線ツリーのPEは幾可的に最も近い他のPEに連結されていない。これでは、直接通信経路が存在しないので、多く

の重要な操作を効率良く行うことができない。事実、3×3のウィンドウ内の半分を越えるデータはツリーの1レベルからウィンドウの中央へと移動しなければならない。3×3ウィンドウのデータ間の平均的距離は2．125であるが、多形網目では1．5である。直角線ツリーの3×3のウィンドウの例は第8図及び第10図に示されており、丸印に付された数字はデータとウィンドウの中心との距離を表わしている。

N＝4の場合の第7図の例について述べると、多形網目と直角線ツリーとでは、チップ面積比は16：46であり、3×3ウィンドウの平均的距離の比は1．5：2．1（第8図及び第10図）である。

制御アルゴリズムORTH＿TREEは、BSRX内でタイムステップtにおいて、レジスタXをpid0に、SRXをpid0の複写に、F1をhマスクに用いてpid0‹t›をつくり出す。同様に制御アルゴリズムは、BSRY内でtime step tにおいて、レジスタYをpid1に、SRYをpid1の複写に、F2をレマスクに用いてpid1‹t›をつくり出す。条件付SHORT＿WE及びSHORT＿NSはBSRX、BSRY、F1、F2に基礎を置いている。

（e）パターン5（逆一行ツリー）

RRツリー（Revese-Rowツリー）は頂部から低部へとデータが流れるツリーである（行ツリーは底部から頂部へと流れる。）。制御アルゴリズムは次のようである。

```
RR－TREE()
(
int t; /*t is time step*/
int pid0; /*column position of a PE*/
int M, logM; /*M is the side size of the
mesh and logM＝logM*/
int treemask＝M%2; /*a flag to construct
the tree*/
int mask; /*an intermediate condition*/
  for(t＝0; t<logM; t++)(
    mask＝ANDALLBIT
        (treemask | pid0);
    if(¬mask)
      (SHORT_WE; DISABLE; )
    if(mask && pid0<logM-t-1>)
      W＝MEM;
    if(mask && ¬pid0<logM-t-1>)
      MEM＝E;
    treemask＝ASHIFT(treemask, 1);
  )
)
```

第11図の8－PEの例については制御アルゴリズムを次のように説明することができる。

フラグツリーマスクはPEの総数の半分について初期化される（即ち、4＝100）。INVERTされたツリーマスクはまずpid0とORされてその結果はANDALLBITに渡され、ANDALLBITはもしその結果の全ビットが1であれば1をリターンし、その他の場合では0をリターンする。マスク＝1のPE（即ち、PE3及びPE7）はツリーの一部であり、ツリーの節でない他のPEは自らを使用禁止にするかあるいはツリー連結を確立するためにW－EバスをSHORTさせる。PE3及びPE7では、pid0のビット2が更にチェックされ、それが1であればPE7をしてデータをPE3に送らせる（それまではPE3のビット2は0である。）。これはt＝0においてツリーのトップレベルを形成する。

t＝0の最後にツリーマスクは算術的に左へ1ビットシフトされる。したがって、次のタイムステップ用110となる。

t＝1では、PE1、3、5、7がツリー節であり、PE7がPE5に、また、PE3がPE1にデータを送る。t＝1の最後にツリーマスクは111になる。

t＝2では全PEがツリー節となり奇数pid0の各PEは偶隣接する低位の偶数pid0の各PEにデータを各々送る。

ＣＣＭ中の機構を用いて、ツリーマスクはＳＲＭにロードされ、ＸはＳＲＸに複写される。レジスタＸはＩＮＶＥＲＴされたＳＲＭとＯＲされ、次に、その結果はマスクをつくるためにＡＮＤＡＬＬＢＩＴされる。ＳＲＸはタイムステップｔにおいてＢＳＲＸ内にて左へ論理的にシフトされてpid０＜logＭ－ｔ－１＞をつくる。これはツリー節のペア用にＳＥＮＤ／ＲＥＣＥＩＶＥを制御するように用いられる。

ＲＲツリーはデータを全ツリー節に拡げるためのものであり、全ツリー節はツリー内の位置に応じて異なる操作を施す。計算機図形処理では、このパターンは有用である。というのは、Ａを定数Ｘをpid０として各ＰＥにＡ×Ｘを同時に行なわせるからである。Ａ×Ｘを並列的に調べることによりラインの最初の生成を行うことができる。これについては、対角線拡りツリー（パターン１２）について述べるところで更に説明する。

一般に逆ツリーはパラメータ空間の記号表現を画像空間の画像表現に変換するために有用であり、アルゴリズムは多形網目による非常に大量の並列処理を行う。

制御アルゴリズムは最高位のpid０を有するＰＥをツリーの根として用いるが、最低位のpid０を有するＰＥもツリーの根として用いることもでき、制御アルゴリズムは同等の複雑さを有する。

(f) パターン６（逆列ツリー：ＲＣツリー）

ＲＲツリーと同様に、ＲＣツリーもpid１を制御子とし、Ｎ－Ｓをツリー確立のための経路とすることにより形成される。制御アルゴリズムは次のとおりである。

```
RC-TREE()
{
int t; /*t is time step*/
int pid1; /*row position of a PE*/
int M, logM; /*M is the side size of the mesh and logM = logM*/
int treemask = M$2; /*a flag to construct the tree*/
int mask; /*an intermediate condition*/

  for(t=0; t<logM; t++){
     mask=ANDALLBIT
                    ( treemask | pid1);
     if(¬mask)
        {SHORT_NS; DISABLE;}
     if(mask && pid1<logM-t-1>)
         N=MEM;
     if(mask && ¬pid1<logM-t-1>)
         MEM=S;
     treemask=ASHIFT(treemask, 1);
  }
}
```

ＲＣツリーの特性はＲＲツリーと同様であり、ただ、ＲＣツリーでは網目の列（コラム）中のデータに関連している点で異なるだけである。

(g) パターン７（行バス）

同報通信目的では、バスは同報通信距離を最短にするので非常に有用である。１つのバスが以下の制御アルゴリズムにより多形網目の全ての行について形成される。

```
ROW_BUS()
{
int sender; /*ID for the sender*/
int pid0;

  SHORT_WE;
  if(pid0==sender)
     E=MEM;
  else
     MEM=W;
}
```

行中の１のＰＥが送り元として働き、残りのＰＥは受け手となる。全てのＰＥはＥ－ＷをＳＨＯＲＴさせてバスを確立し、送り元はデータをＥ（あるいはＷ）に送り、受け手はデータをＷ（あ

るいはE）から受け取る。この他に、データはWあるいはEに外部コントローラから注入され、送り元としてのPEは存在せず、全てのPEが受け手である場合もある。

ＣＣＭにより送り元はＳＲＭにロードされる。ＩＮＶＥＲＴされたＳＲＭはpid０を格納しているレジスタＸとＸＯＲされる。得られたビット値はＡＮＤＡＬＬＢＩＴされる。ＸＡＮＤＡＬＬ中の１は送り元としてのＰＥを特定し、ＸＡＮＤＡＬＬ＝０のＰＥは受け手となる。

(h) パターン８（列バス）

行バスと同様に、列バスも、pid１を制御子とし、Ｎ－Ｓを経路とすることにより、多形網目の各列について形成される。制御アルゴリズムは次のようである。

```
COLUMN_BUSO
{
int sender; /*ID for the sender*/
int pid1;
 SHORT_NS;
 if(pid1==sender)
   S=MEM;
 else
   MEM=N;
}
```

列バスの特性は行バスと同様である。

行バスと列バスとが組み合わせられることにより２ステップで列データが全ＰＥに同報通信される。第１ステップでは、列データが最上位の行の全ＰＥに同報通信され、次に第２ステップでは、最上位の行のＰＥは他の全ＰＥに列方向に沿って列データを同報通信する。

(i) パターン９（ピラミッド）

ピラミッド構造は、多重解像度画像を取扱う上で優れている。ピラミッドパターンは次の制御アルゴリズムにより多網目から形成される。

```
PYRAMIDO
{
int t; /*t is time step*/
int pid0; /*column position of a PE*/
int pid1; /*row position*/
int M, logM; /*M is the side size of the
mesh and logM=logM*/
int hmask=1, vmask=1; /*two flags to
construct the pyramid*/

  for(t=0; t<logM; t++){
  /*cycle 1 action*/
     if(¬hmask | ¬vmask)
        {SHORT_WE; SHORT_NS;
         DISABLE;}
     if(hmask && vmask && ¬pid0<t> &&
        ¬pid1<t>)
         E=MEM;
     if(hmask && vmask && ¬pid0<t> &&
        ¬pid1<t>)
        {N=MEM; MEM1=W;}
     if(hmask && vmask && ¬pid0<t> &&
        pid1<t>)
         E=MEM;
     if(hmask && vmask && pid0<t> && pid1<t>)
        {MEMO=N; MEM2=W;}

  /*cycle 2 action*/
     if(¬hmask | ¬vmask)
        {SHORT_WE; SHORT_NS;
         DISABLE;}
     if(hmask && vmask && ¬pid0<t> &&
        ¬pid1<t>)
         NO_ACTION;
     if(hmask && vmask && pid0<t> &&
        ¬pid1<t>)
         S=MEM1;
     if(hmask && vmask && ¬pid0<t> &&
        pid1<t>)
         NO_ACTION;
```

```
    if(hmask && vmask && pid0<t> && pid1<t>)
        MEM1=N;

    hmask=hmask & pid0<t>;
    vmask=vmask & pid1<t>;
  }
}
```

制御アルゴリズムはlogMのステップを含み、各ステップには２つの制御サイクルが存在している。別言すれば、ステップは２つのＰＥサイクルでピラミッドの１つのレベルを形成する。

第１２図、第１３図、及び第１４図は８×８網目におけるピラミッド制御アルゴリズムを示している。

ｈマスク（行用）及びＶマスク（列用）の２つのマスクが１に初期化され網目中の全ＰＥが初めのステップにおいて使用可能になる。ｔ＝０Ｚおいて、全ＰＥが処理操作状態（アクティブ状態）となり、２×２のＰＥが１つのグループを形成する。これら４つの（２×２）ＰＥは、ピラミッドのＮＷ、ＮＥ、ＳＷ、及びＳＥ息子であり、親はＳＥ息子と同様である。４つの息子の処理操作はpid０＜ｔ＞及びpid＜ｔ＞のビット値で区別される。pid０＜０＞＝pid１＜０＞＝１のＳＥ息子（あるいは親）は、第１サイクルにおいて、pid０＜０＞＝０且つpid１＜０＞＝１のＳＷ息子及びpid０＜０＞＝１且つpid１＜０＞＝０のＮＥ息子からデータを受け取る。このサイクル中で、ＮＷ息子はデータをＮＥ息子からデータを受け取る。このサイクル中で、ＮＷ息子はデータをＮＥ息子へと渡す。このデータは第２サイクルで親により受取られる。第２サイクルでは、ＮＥ息子と親との間だけでデータの受渡しが行なわれ、他の２のＰＥは活動しない。Ｖマスクとｈマスクは次のタイムステップの連結を制御するために更新される。

ｔ＝１において、再び４つのＰＥがピラミッドの次のレベルの４つの息子と親とを形成する。４×４網目中のこれら４つのＰＥの幅は第１３図に示されている。４つの息子と親の処理操作はｔ＝０で等しいが、偶数行と偶数列のＰＥは使用不可である。これらの使用不可の非ピラミッドＰＥは自らのＷとＥライン及びＮとＳラインをＳＨＯＲＴしてピラミッド連結を確立させる。

第１４図にはピラミッドの最終レベルを構成するＰＥが示されている。

上述の直角線型ピラミッド構造では、ピラミッドパターンは各レベルに網目連結を有している。これは同一レベルに４つの近傍（Ｎ、Ｓ、Ｅ、Ｗ）が存在しているピラミッド中の各ノードであり、下のレベルの４つ息子や上のレベルの１つの親のためのものではない。この関係は第１５図に示されている。

同一レベルの近傍ＰＥを得るための制御アルゴリズムは上述の制御アルゴリズム中に組み込まれている。例えば、第１５図で、ｔ＝０では、同一レベルの近傍ＰＥは元の網目に連結されている。ｔ＝１では、同一レベルの近傍ＰＥは他の行及び列に拡がり網目連結が確立される。

ピラミッドの同一レベルでの近傍ＰＥの内容を得るためには、下記の制御アルゴリズムＰＹＲＡＭＩＤの各ステップに２つの制御サイクルが付加される。サイクル３はＮとＷの内容を得るため、サイクル４はＳとＥの内容を得るためのものである。

```
PYRAMIDO
(
int t; /*t is time step*/
int pid0; /*column position of a PE*/
int pid1; /*row position*/
int M, logM; /*M is the side size of the
mesh and logM=logM*/
int hmask=1, vmask=1; /*two flags to
construct the pyramid*/

  for(t=0; t<logM; t++)(
  /*cycle 1 action*/
     if(¬hmask | ¬vmask)
       (SHORT_WE; SHORT_NS;
```

```
      DISABLE; }
   if(hmask && vmask && ¬pid0<t> &&
      ¬pid1<t>)
      E=MEM;
   if(hmask && vmask && pid0<t> &&
      ¬pid1<t>)
      {N=MEM; MEM1=W; }
   if(hmask && vmask && ¬pid0<t> &&
      pid1<t>)
      E=MEM;
   if(hmask && vmask && pid0<t> && pid1<t>)
      {MEM0=N; MEM2=W; }

/*cycle 2 action*/
   if(¬hmask | ¬vmask)
      {SHORT_WE; SHORT_NS;
       DISABLE; }
   if(hmask && vmask && ¬pid0<t> &&
      ¬pid1<t>)
      NO_ACTION;
   if(hmask && vmask && pid0<t> &&
      ¬pid1<t>)
      S=MEM1;
   if(hmask && vmask && ¬pid0<t> &&
      pid1<t>)
      NO_ACTION;
   if(hmask && vmask && pid0<t> && pid1<t>)
      MEM1=N;

/*cycle 3 action*/
   if(¬hmask | ¬vmask)
      {SHORT_WE; SHORT_NS;
       DISABLE; }
   if(hmask && vmask){
       S=MEM3;
       E=MEM4;
       MEM3=N;
       MEM4=W;
   }

/*cycle 4 action*/
   if(¬hmask | ¬vmask)
      {SHORT_WE; SHORT_NS;
       DISABLE; }
   if(hmask && Vmask){
       N=MEM5;
       W=MEM6;
       MEM5=S;
       MEM6=E;
   }

   hmask=hmask & pid0<t>;
   vmask=vmask & pid1<t>;
 }
}
```

ＣＣＭの働きにより、ｈマスク及びＶマスクがＦ１及びＦ２に夫々ロードされる。レジスタＸ内のpid０及びレジスタＹ内のpid１はＳＲＸ及びＳＲＹに夫々複写される。右に論理的にシフトすることにより、ＢＳＲＸ及びＢＳＲＹはpid０＜ｔ＞及びpid１＜ｔ＞を、タイムステップｔにおいて、その内容とする。Ｆ１及びＦ２とともにこれらの２つの条件はＳＨＯＲＴ処理操作の実行に用いられる。

上述のピラミッドはＭ×Ｍのベースを有し、縮退２である。縮退２とはＭ×Ｍのベースの上部レベルが（Ｍ／２）×（Ｍ×２）であり、以下これを繰り返すことを意味する。ＰＹＲＡＭＩＤ制御アルゴリズムはＫを２の累乗数として縮尺Ｋの場合に拡張できる。そのためには、以下のようにしてｈマスク及びＶマスクを更新すればよい。

ｈマスク＝ｈマスク ＆ pid０＜ｔ＞＆ pid０＜ｔ＋ｉ＞

Ｖマスク＝Ｖマスク ＆ pid１＜ｔ＞＆ pid１＜ｔ＋ｉ＞

そして、ピラミッドノード処理操作を偶数ｔステップ毎にスキップすればよい。

(j) パターン10（逆ピラミッド）

画像データから記号（符号）データへの変換にはピラミッドの底から頂部へと情報が流れる。しかし、記号データから画像データへの変換には逆方向に情報を流す必要がある。これは、以下の制御アルゴリズムにより多形網目から形成される逆ピラミッド（Rピラミッド）により実行される。

```
R－PYRAMIDO
{
int t; /*t is time step*/
int pid0; /*row position of a PE*/
int pid1; /*column position of a PE*/
int M, logM; /*M is the side size of the
mesh and logM＝logM*/
int mask＝M%2; /*a flag to construct the
pyramid*/
int hmask, vmask;

  for(t＝0; t<logM; t++){
    hmask＝ANDALLBIT(¬mask | pid0);
    vmask＝ANDALLBIT(¬mask | pid1);

  /*cycle 1 */
    if(¬hmask | ¬vmask)
      {SHORT_WE; SHORT_NS;
       DISABLE; }
    if(hmask && vmask && pid0<logM-t-1> &&
      pid1<logM-t-1>)
       N＝MEM2;
    if(hmask && vmask && pid0<logM-t-1> &&
      ¬pid1<logM-t-1>)
       MEM2＝S;
    if(hmask && vmask && ¬pid0<logM-t-1>
      && pid1<logM-t-1>)
       NO_ACTION;
    if(hmask && vmask && ¬pid0<logM-t-1>
      && ¬pid1<logM-t-1>)
       NO-ACTION;

  /*cycle 2 */
    if(¬hmask | ¬vmask)
      {SHORT_WE; SHORT_NS;
       DISABLE; }
    if(hmask && vmask && pid0<logM-t-1> &&
      pid1<logM-t-1>)
      {N＝MEM1; W＝MEM3; }
    if(hmask && vmask && pid0<logM-t-1> &&
      ¬pid1<logM-t-1>)
      {MEM1＝S; W＝MEM2; }
    if(hmask && vmask && ¬pid0<logM-t-1>
      && pid1<logM-t-1>)
       MEM3＝E;
    if(hmask && vmask && ¬pid0<logM-t-1>
      && ¬pid1<logM-t-1>)
       MEM2＝E;

    mask＝ASHIFT(mask, 1);
  }
}
```

R_PYRAMID制御アルゴリズムはPYRAMIDアルゴリズムの逆方向処理であり、RR_TREE及びRC_TREEの拡張である。第19図に簡単な例が示されている。この例では8×8網目中の3つのレベルが示されている。

第19図に示されるように、逆ピラミッドの根はPE（0、0）（図中左上端のPE）である。その左息子はPE（4、0）、右息子はPE（0、4）である。

第2レベルの息子は横方向及び縦方向の第1レベルの息子の夫々に関して2つある。PE（6、0）及びPE（2、4）がPE（0、4）の息子であり、PE（4、2）及びPE（6、0）がPE（4、0）の息子である。

同様にして第3レベルの息子も定まる。

一般的に言って、N×N網目ではPE（0、0）が根であり、K＝0からN－1であるとしてPE（k、N－1－k）がリーフである。PE（s、t）の第i番目（i＝1からlogNとする）のレベル左息子はPE（S＋2（logN－i）、t）

であり、右息子はＰＥ（ｓ、ｔ＋２（ｌｏｇＮ－ｉ））である。

制御アルゴリズムは次のようである。

```
ULDST0
(
int fs=0, fr=0; /*flag-send is used to
                  construct the DST*/
                /*flag-receive is an
                  intermediate var to
                  update fs*/
int pid0, pid1;
int t; /*t is time step*/
int M, logM; /*M is the side size of the
mesh and logM=logM*/
int treemask=M$2; /*a flag to construct
                     the tree*/

   if(pid0==0 && pid1==0)(
      fs=1; fr=1; )
  for(t=0; t<logM; t++)(
     hmask=ANDALLBIT
          (¬treemask | pid0);
     vmask=ANDALLBIT
          (¬treemask | pid1);
     if(¬hmask | ¬vmask)
       (SHORT_WE; SHORT_NS;
       DISABLE; )
     if(hmask | vmask && fr)
       (E=fs; S=fs; fr=0; )
     if(hmask | vmask && ¬fr)
       (fs=W | N; fr=W | N; )
     treemask=ASHIFT(treemask, 1);
  )
)
```

８×８の多形網目の例で上記アルゴリズムを説明する。ＰＥ（０００、０００）をＤＳＴ（Diagonal-Span-Tree：対角線パスツリー構造）の根として選び、fs（flag send：フラグ送り）及びFr（flag receive：フラグ受け取り）を１にセットする。ｔ＝０のタイムステップで、行０００と１００、及び列０００と１００がアクティブであり、残りはディスエイブル（不活動）である。不活動なＰＥはＷＥ及びＮＳを短絡してfsを更新するために新しいパスを確立する。fr＝１なるfrを有するアクティブなＰＥはそのfs値をＥ及びＳ近傍へ送り、次のタイムステップでは送り手とならないようにfrを０にリセットする。受け手（fr＝０である。）はそのfsをＮとＷのＯＲされた値として更新する。受け手は又そのfrを同じＯＲされた値として更新し、それゆえ、Ｎ又はＷから１を受け取ったＰＥは次のタイムステップでは送り元となる。ステップｔ＝０では、２つのＰＥ（ＰＥ（０００、１００）とＰＥ（１００、０００）のこと。）がＤＳＴのノードとして（fs＝１にセットすることにより）選択され、更に、これらが新しい送り元として（fr＝１とセットすることにより）準備され、後続のステップにおいて更なるＤＳＴがセットされる。

次のステップでは、先の２つの新しい送り元の各々が２つのＤＳＴノード及び２つの送り元を同様の方法で形成する。新しいノード及び送り元はＰＥ（０００、１１０）、ＰＥ（０１０、１００）、ＰＥ（１００、０１０）、及びＰＥ（１１０、０００）である。

ステップｔ＝２では、対角線ＰＥが制御アルゴリズムにより達成される。それらのfsが１にセットされて自らがＤＳＴの部分であることが示される。更に、それらのfrが１にセットされ、自らが対角線ノードであることが示される。対角線の識別はＤＳＴアルゴリズムによる付加効果である。このことは、次のセクションで説明する種々の計算に役立つ。

ＤＳＴを形成するため、フラグfsは条件フラグとして用いられる。fs＝０のＰＥはＷＥ及びＮＳパスを短絡させ、fs＝１のＰＥはメモリ内容をＥ及びＳに送るとともにＷ及びＮからデータを受け取る。

連結制御ブロックの機能を利用して、“ツリー

マスクがＳＲＭへ、fsがＦ１へ、frがＦ２へ、各々ロードされる。ＩＮＶＥＲＴされたＳＲＭの内容は、レジスタＸ内のpid０及びレジスタＹ内のpid１の各々とＯＲされる。このようにＯＲされた結果は“ＡＮＤＡＬＬＢＩＴ”されてＸＡＮＤＡＬＬ内に“ｈマスク”を、ＹＡＮＤＡＬＬ内に“Ｖマスク”を生成させる。各ステップの最後において、ＳＲＭが右へ自動的にシフトされる。ＳＨＯＲＴ動作は、こうして、Ｆ２、ＸＡＮＤＡＬＬ、及びＹＡＮＤＡＬＬに基づく。

異なる根を選択して同様の制御アルゴリズムを用いることにより、同じ多形網目中に異なるＤＳＴが形成され得る。第２０図には下の右端に根を有するＤＳＴが示されている。これをＵＲＤＳＴという。

以下において、ＵＬＤＳＴとＬＲＤＳＴとを共存させてＡ・Ｘ＋Ｂ・Ｙ＋Ｃという計算を画像の各画素（ｘ、ｙ）について並行処理により行え得ることを示す。このような能力は計算機図形（画像）処理に関して幅広く役立つ。

Ｅ－４．アプリケーション

単純な網目の画像処理へのよく知られたアプリケーションの他に、多形網目の下記のアプリケーションは、単純な網目では実行不可能であったり或いは単純な網目より高速である。

これらのアプリケーションは６つの型に分類できる。

（1）分割統合処理

このタイプの計算には、Ｎデータの１つのセットをそれらの性質に応じて２つのグループに分ける計算が含まれる。次に同様にして各グループは２つのサブグループに分けられる。このような処理がグループ内のデータが唯１つになるまで繰り返される。

網目連結では、このタイプの計算はＯ（Ｎ）あるいはそれ以上の計算量になる。しかし、多形網目においてツリーあるいはピラミッドに変形すれば、このタイプの計算はＯ（logＮ）の計算量になる。Ｎ＝１０２４のデータセットであれば、１００：１の比の（つまり、２桁の）スピードアップが図られる。

このタイプに属する計算は、分類、最大値発見、最小値発見、Ｋ番目の大きさの値の発見、中間値の発見等の計算である。これら全てのアルゴリズムの計算量（計算ステップ数）はＯ（logＮ）である。

（2）画像データと記号データ間の変換

このタイプの計算は計算機図形処理に特有のものであり、しばしば中間レベル処理と呼ばれる。ある画像が与えられたとき、次の事を検討しなければならない。

（a）どれ位の画素が特定の性質を満足するか。

（b）どの画素が特定の性質を満足するか。

（c）幾つかの画素あるいは全ての画素が特定の性質を満足するのか、あるいは特定の性質を満足する画素は無いのか。

ここで、上記特定の性質とは、ある値と等しいとか、ある値より大きいとか、ある値より小さいとか、あるいはこれらから算術的または自動的に合成される条件である。

これらの全てのアルゴリズムは多形網目におけるツリーやピラミッドパターンによりＯ（logＮ）ステップで計算される。より重要で従来の固定的パターンと比較されるべき特徴は出力に関するものである。多形網目ではＩ／Ｏ速度が向上する。多くの場合、１０２４×１０２４ビット（全画像）に対して唯の１ビット（ハイ／イイエ）が出力されるだけである。

Ｉ／Ｏに関連して、余分のＮ－Ｓパスが１／Ｏ及び処理の同時流れのために網目に付加されたことがある。このような機構及び利点は多形網目にも有効であるが、本発明の必須とするところではない。

（3）統計

多形網目は以下の統計的計算をＯ（logＮ）ステップで行うことができる。

（a）平均、差異、標準偏差

（b）面積、周辺長さ、図心

（c）第１モーメント、第２モーメント、クロスモーメント

上記(a)はNデータ一般に関するもので、(b)及び(c)は画像に特有のものである。

統計的計算は他のアルゴリズムの基礎である。計算機図形処理において統計的計算は領域解析やパターン認識の基礎である。

(4) A・x+B・y+Cの計算

A・x+B・y+Cを計算するためには、多形網目による4つのパターンが必要となる。それらは上方左対角スパンツリー（ULDST：Upper-Left-Diagonal-Span-Tree）、下方右対角スパンツリー（LRDST：Lower-Right-DST)、行バス、及び列バスである。ULDSTとLRDSTとはA・x+C及びB・yを同時に計算するために共存する。行バスと列バスは合算（例えばA・x+C+B・y）を行うために共存する。

アルゴリズムはビット連続的に実行される。画素平面内の2つの特殊なツリーが消去され得る。定数A及びBが計算開始前に全PEに同報通信され、アレイの開始部分に先取りされている。(logM－1）の0とともにアレイA及びBにストアされる。ストアされたA及びBはビット反転され、先取りされた0がアクセスされた後に、A及びBの下位桁側のビットが先ずアクセスされるようになっている。定数CはULDSTの根のWを通じて最下位ビットからスタートして1タイムステップ当り1ビットずつ多形網目に注入される。A・x+Cを計算するためにULDSTをツリーとして用いるとき、各PEは3つの変数を有する（合算、キャリー、及び遅延）。各タイムステップにおいて、合算は、東境界へ渡され、遅延は南境界側へ渡され、各PEは、(a)NにアレイAを加え、キャリービットをキャリーにストアし、(6)Nを遅延にストアするという2つの処理を行う。logMステップの後、網目の対角線PE（あるいはULDSTのリーフ）が対応する行のためにA・xをストアする。同様に、B・yの計算は、根のEから0が注入され且つPEが遅延を北境界側に、合算を西境界側に渡すようなLRDSTにより実行される。logMステップの後、各対角線PEは対応する列のためにB・yをストアする。

行中にA・xを、列中にB・yを得た後、多形網目はWEバスの行バスとNSバスの列バスに変形する。各PEは行バス上の値と列バス上の値とを加えてA・x+B・y+Cをビット連続形式で算出する。

DSTとバスとを同時に確立することができないため、A・x+B・y+Cの結果は他の毎回のタイムステップ毎にビット連続的に伝えられる。

(5) 第1ライン検出

多形網目の2つのタイムステップ毎にA・x+B・y+Cを計算する能力により、M×M画像中の画素（x、y）がA、B、Cで定まる与えられたあるライン上にあるか否かを決定できる。A、B、Cの全ての数がKビット長であるとき、(logM+2K）タイムステップで決定が行なわれる。

第1ライン検出機能は計算機図形処理及び計算機視覚処理に非常に有用である。計算機図形処理にとっては、凸形多角形の表示、影の形成、クリッピング、球の描画、ヒストグラムの同等化計算、テキスチャのマッピング、及び非変名処理（anti-aliasing）に役立つ。計算機視覚処理では、ノイズ画面中にラインを見つけるための第1Hough変換を計算する上で役立つ。

(6) 記号情報の画像情報への変換

多形網目による大規模な並列処理ハードウェアを用いると、記号処理（通常は網目では実行されない）から画像処理への変換して処理を大規模な並列処理により実行してしまうことができる。上述の第1Hough変換はそのような一例である。マーク発生は本カテゴリー中の他の例である。

(6．1) バンドマスク発生

バンドマスクは2平行ライン以内に制限され、その一方は（A、B、C1）により、他方は（A、B、C2）により夫々決定される。バンドマスクを発生するため、各PEはA・x+B・y+C1及びA・x+B・y+C2を上述のように計算する。計算結果はS1（A・x+B・y+C1の符号）及びS2（A・x+B・y+C2の符号）であり、これらは画素（x、y）がバンドの内側に存るか否かを決定するために用いられる。

バンドマスクは、計算機視覚処理において、興味ある領域内の処理だけを行うために役立つ。人間の視覚はマスクを発生するには異なる方法を採っている。その方法とは記号情報に関するが、その処理は実際には上述のように画像的に行なわれる。

(6.2) 多角マスク発生

多角マスクはバンドマスクを一般化したものである。それはP部分平面から成り、P部分平面の各々はA・x+B・y+Cで特定されるラインにより決定される。ライン検出機能を用いると、対応するラインについて符号S1、S2……SPを得ることができる。S1からSPのブール結合による画素(x、y)がその多角形内側に存るかどうかが決定される。

E-5. 結論

連結制御機構(CCM)の制御により次の変換操作が可能である。

物理的なM×M網目は、M×Mの長さの1つの行及びM×Mの長さの1つの列に連結できる。

物理的なM×M網目は、各々がMのリーフを有するM行ツリーに連結できる。

物理的なM×M網目は、各々がMのリーフを有するM列ツリーに連結できる。

物理的なM×M網目は、M×M直交ツリーに連結できる。

物理的なM×M網目は、各々がMのリーフを有するM逆行ツリーに連結できる。

物理的なM×M網目は、各々がMのリーフを有するM逆列ツリーに連結できる。

物理的なM×M網目は、各々がMのPEを有するM行パスに連結できる。

物理的なM×M網目は、各々がMのPEを有するM列パスに連結できる。

物理的なM×M網目は、M×Mのベースを有し縮小率K(Kは2の累乗数)のピラミッドに連結できる。

物理的なM×M網目は、M×Mのベースを有し縮小率K(Kは2の累乗数)の逆ピラミッドに連結できる。

物理的なM×Mの網目は、M×M×K(KはPEのローカルメモリで制限される整数)の立方体に連結できる。

また、CCMの外部のプログラムの制御下で以下の変換が行なわれる。

物理的なM×Mの網目はDSTツリー構造に連結できる。DSTツリーの根は網目のどこの角にあってもよい。根が対角線の反対方向の角にあれば、2つのDSTが共存できる。

物理的なM×Mの網目は、O(M^2)のシリコン上の面積内にM×Mの直交線ツリーに連結できる。本発明によれば、O[$(\log M)^2$]の要因により、面積が節約される。なお、Mは直交線ツリーの側辺の大きさである。

分割統合アルゴリズムは、O(M)という線形の計算量(計算ステップ数)の場合とO(logM)という対数形の計算量の場合とに分類される。M/logMの節約が本発明により得られる。これについても既に実施例の説明で述べられている。

画像から記号への変換(中間レベル処理)操作は多形網目内部で行なわれ、I/O量は非常に減小される。非常に多くの場合、6桁程度の減小が図られる。

記号表現から画像表現への変換が上述のパターンにより行なわれて画像に関する処理が大規模に並列的に行なわれ得ることになる。そのような構成を記号処理の範囲の網目に拡張してもよい。

網目は次のような機能を有する。

(a) 画像処理を行う。

(b) 画像データを記号データに変換する。

(c) 記号データを画像データに変換する。

(d) 記号データ処理をその画像データ等価物において行う。

本発明によれば、M×M網目内でO(logM)ステップにて、各画素(x、y)に対し並列に、A・x+B・y+Cの計算ができる(A、B、Cは定数とする。)

本発明によれば、各画素(x、y)が(a)ライン上か、(b)ラインの右側か、(c)ラインの左側かの検出をO(logM)計算ステップ数で行うこと

ができる。

本発明によれば、各画素(x、y)が2つの平行線で形成されるバンドの内側か外側かの検出を並列的に行うことができる。

本発明によれば、各画素(x、y)が多角形の内側か外側かの検出を並列的に行うことができる。

本発明は、レジスタZ、シフトレジスタSRZ、及びフラグレジスタF3をCCMに付加することにより、3次元網目(物理的立方体)に形成して、12のパターンのより高次元の拡張を果すことができる。

3次元(3D)画像素子はボクセル(voxel)である。ボクセルは面積のみを有するピクセル(画素)と類似している。本発明は3次元に拡張して各ボクセル(x、y、z)が(a)2つの平行平面で形成される領域、(b)多角形体内に存るか否か、あるいは(c)ボクセルがある平面上か、左か右かについての検出を並列で行うことができる。

本発明は多形という概念を物理的網目にCCMを通じて適用している。同様の概念及び機構を他の固定的な連結構造に適用することもできる。

多形網目により形成されるパターンは、ALUの出力を通じてF1及び/またはF2にロードされる値の内容により、データの性質に適するようにされる。

任意のパターンが、命令又はメモリ内容を通じてF1及びF2レジスタをセットすることにより、多形網目hから形成され得る。命令、メモリ内容、近傍PEからの中間処理値、及びPE内部の診断情報は、典型的システム操作パラメータであり、これらはよく知られた技術及び簡単な手段によって、フラグレジスタのセットに用いられる。CCMはシステム操作パラメータによりセットされ得るフラグレジスタ手段を有し、パターンレジスタの少なくとも1つに新しいパターンをセットするために用いられ得る制御情報を発生する。このような機能は、プログラマによりアクセス可能であり、プログラマによりデータに関して或いは条件に関して更に他の事象取扱いの可能性を有する。そのような事象では、フラグレジスタがセットされ、それに応答して新しいパターンがフェッチあるいは計算される。

4．図面の簡単な説明

第1図は本発明で用いられる複数の多形網目処理要素を有する網目組織から成る画像処理装置を示すブロック図、

第2図は前記多形網目処理要素の連結制御機構の切換機構を示すブロック図、

第3図は前記連結制御機構の構成を示すブロック図、

第4図は前記多形網目処理要素が枝分れ等のない一続きのひも状に連結された状態を示すブロック図、

第5図は前記多形網目処理要素が木構造状に連結された状態を示すブロック図、

第6図は前記多形網目処理要素が前記以外の木構造に連結された状態を示すブロック図、

第7図ないし第10図は本発明の多形網目処理要素を用いた場合のチップ上面積の比較及び連絡部の改良状態を示すブロック図、

第11図は前記多形網目処理要素が逆木構造状に連結された状態を示すブロック図、

第12図ないし第20図は各々前記多形網目処理要素の代理的な連結状態を示すブロック図、

第21図は前記多形網目処理要素の内部機構の一例を示すブロックである。

1…アレイ、2…基本演算器、3…ホストコンピュータ、6…ALU、7……メモリ、8…連結制御機構(CCM)、9…バス、10…マトリックス(交換スイッチ)、11…入力線。

出願人　インターナショナル・ビジネス・マシーンズ・コーポレーション
代理人　弁理士　岡田次生
(外1名)

FIG. 1

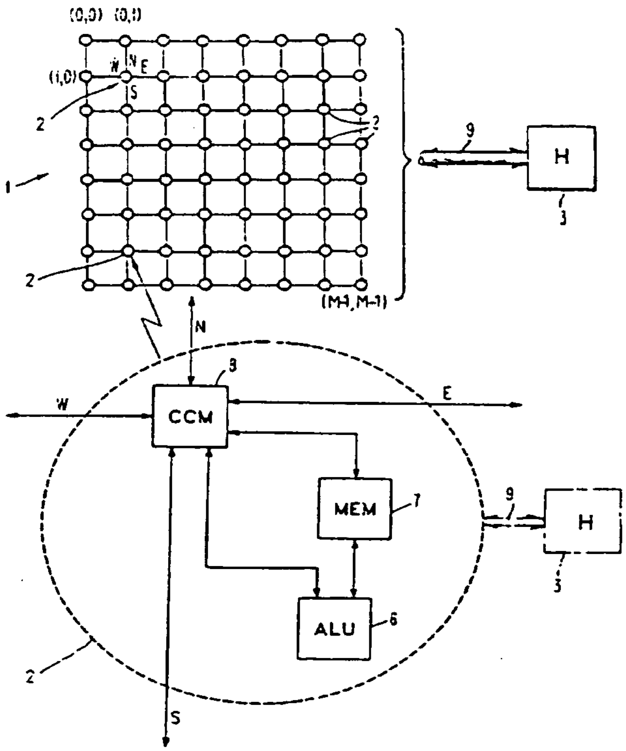

FIG. 2

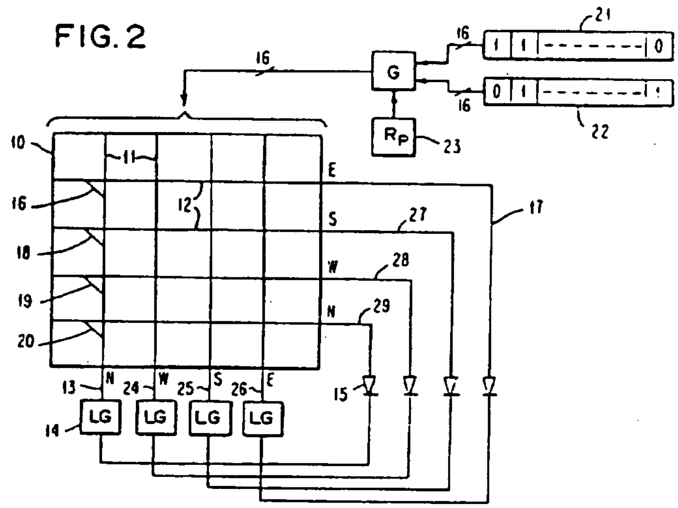

FIG. 3

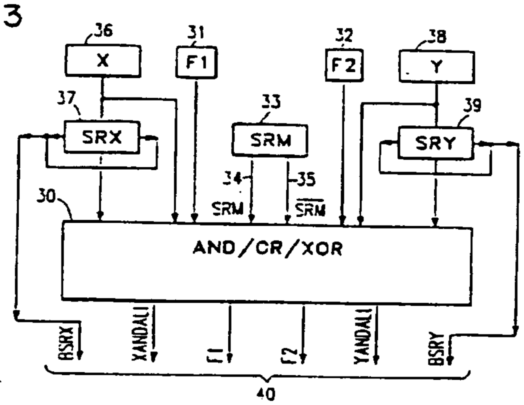

FIG. 4

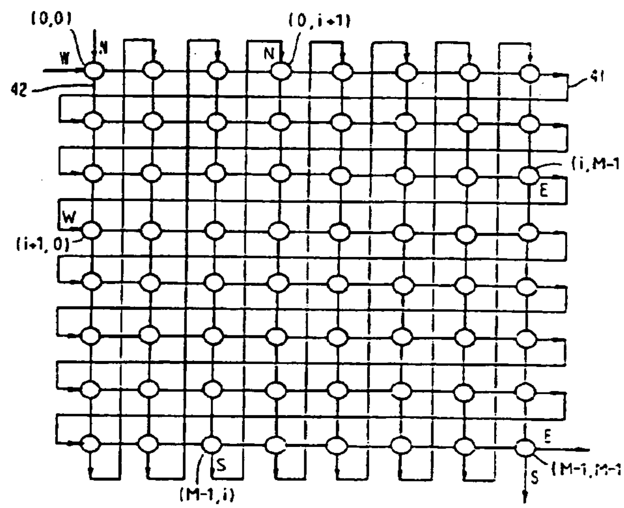

FIG. 7

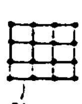

FIG. 9

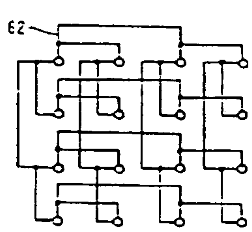

FIG. 5

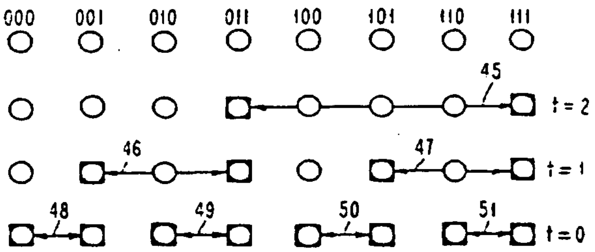

FIG. 8

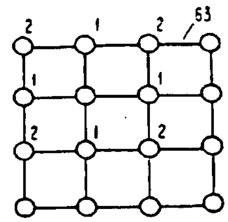

FIG. 10

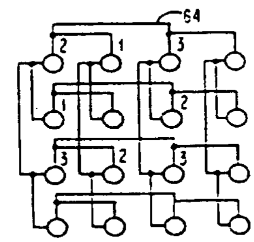

FIG. 6

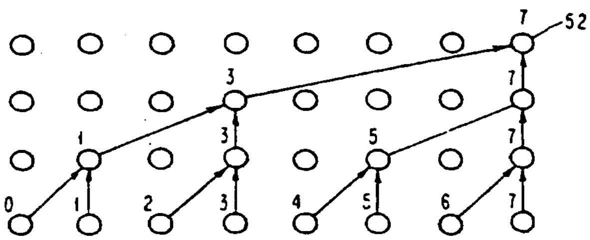

FIG. 11

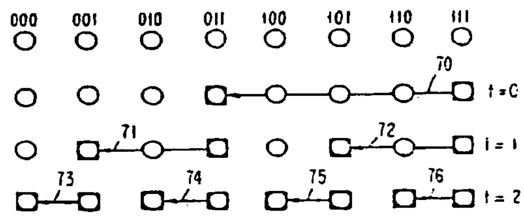

FIG. 12

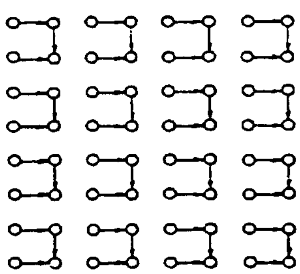

FIG. 14

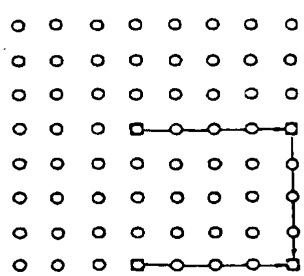

FIG. 16

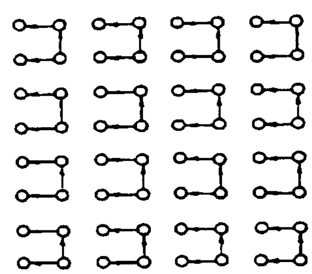

FIG. 17

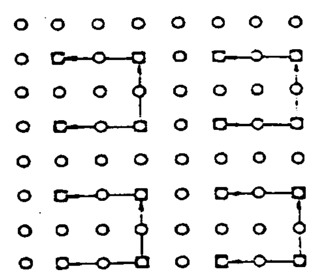

FIG. 13

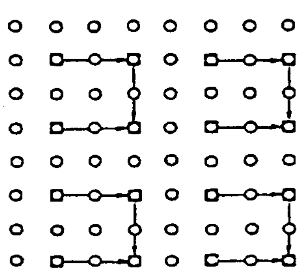

FIG. 15

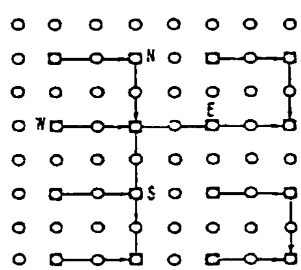

FIG. 18

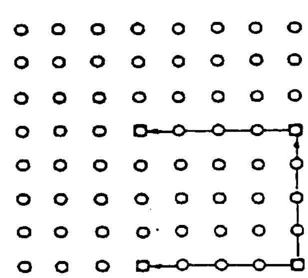

FIG. 19

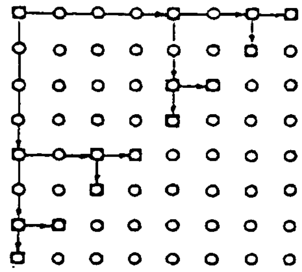

FIG. 20

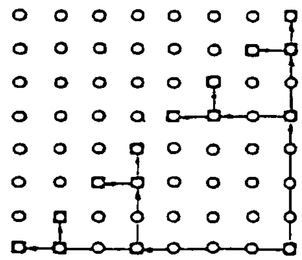

FIG. 21